SPECIFICATION AMENDMENTS

Please replace the paragraph beginning on page 4, line 7 with the following replacement paragraph:

To address the need in the art for a DRM system that meets both consumers and content providers expectations, U.S. ~~Serial~~ Patent Application No. 09/542,510, entitled "Digital Rights Management Within an Embedded Storage Device," filed April 3, 2000, now U.S. Patent No. 6,636,966, U.S. ~~Serial~~ Patent Application No. 09/583,452, entitled "Method of Decrypting Data Stored on a Storage Device Using an Embedded Encryption/Decryption Means," filed May 31, 2000, U.S. ~~Serial~~ Patent Application No. 09/940,026, entitled "Host Certification Method and System," filed August 27, 2001, U.S. ~~Serial~~ Patent Application No. 09/940,083, entitled "A Secure Access Method and System," filed August 27, 2001, now U.S. Patent No. 7,110,982, describe a DRM system in which the DRM "intelligence" has been integrated into the storage engine. As opposed to conventional DRM systems that reside on the host, the integrated storage engine approach is far less vulnerable to hacking by a user of a host system – the user has no access to the DRM functionality within the storage engine other than through the reading or writing of secure content from the storage medium associated with the storage engine. The user knows that digital content may flow to and from the data storage medium but cannot access the "how" within the storage engine that enabled such movement. Moreover, the integration of the DRM system into the storage engine is advantageous in portable applications. Different host systems such as kiosks at a content provider retail outlet or a personal computer may be more readily modified to couple to the portable DRM-system-integrated storage engine.

Please replace the paragraph beginning on page 9, line 10 with the following replacement paragraph:

Turning now to Figure 1, various stages in the authentication process and session key exchange between a host system 5, a hard disc storage engine 20, and a hard disc storage medium 25 are illustrated. Before host system 5 can access or write protected content to disc 25, host system 5 provides storage engine 20 a digital signature 10 from a certifying authority (not illustrated). As discussed previously, storage engine 20 verifies digital signature 10 with the corresponding public certifying authority key. Although host system 5 may carry a valid digital signature 10 from the certifying authority, certain events such as unauthorized access to protected content may lead to revocation of the rights afforded by digital signature 10. To allow for such revocation, disc 25 contains a revocation list 35 in a secure area 40 on disc 25. The revocation list may be stored on the disc during the storage media manufacturing process or updated dynamically through vendor unique commands in the field. Each participant such as storage engine 20 and host system 5 is assigned a unique ID number. Storage engine 20 checks revocation list 35 to verify that the ID number for host system 5 is not on the list. Further details regarding the revocation process are provided in U.S. ~~Serial~~ Patent Application No. 09/940,026, the contents of which are hereby incorporated by reference.

Please replace the paragraph beginning on page 10, line 3 with the following replacement paragraph:

If host system 5 carries a valid digital signature 10 and is not identified on revocation list 35, storage engine 20 may proceed to generate a secure session key 60

using a random number generator 70. Note that because secure session key 60 thus results from a random number generation, it will be unique to each "session" during which host system 5 accesses secure content on disc 25. The duration of a session may be arbitrarily limited to extend just long enough that host system 5 may read or write a single block of secure content from disc 25. Alternatively, the session duration may be extended to cover the access to multiple files or as long as a an implementor deems sufficiently secure. Each participant in a public key security system may have its own public and private key pair. To access secure content on disc 25, host system 5 provides not only digital signature 10 but also its public key 75. Using public key 75, storage engine encrypts secure session key 60 and transmits the encrypted ~~public~~ secure session key to host system 5. Host system 5 may then use its private key 80 to decrypt the transmission to recover secure session key 60.

Please replace the paragraph beginning on page 11, line 24 with the following replacement paragraph:

The integration of these two processes (reading and writing secure content) may be summarized as seen in Figure 2. Both storage engine 20 and host system 5 have their own public key/ private key pairs as discussed above. A session key 60 may be generated by storage engine 20 and then encrypted using the public key 75 from host system 5 as shown in Figure 1. However, it should be appreciated that a similar process may be used wherein an alternative session key is generated using the public key (not illustrated) from storage engine 20. Given that storage engine 20 is the ~~"passive"~~ passive participant in the sense that it is the host system 5 that will initiate requests to

read and write secure content from disc 25, it follows that in response to the requests from host system 5, storage engine 20 would require host system 5 to verify it is authorized to access such content through the generation of secure session key 60 as described previously. In other words, if one participant initiates the need for a verification process, it is the "~~"passive"~~ passive" participant that will generate the corresponding secure session key. Thus, the following discussion will be described with respect to the storage-engine-generated secure session key 60. However, it will be appreciated that the use of a host-system-generated secure session key is also within the scope of the present invention.

Please replace the paragraph beginning on page 13, line 14 with the following replacement paragraph:

During the execution of a SECURE_WRITE_BLOCK command, host system 5 will encrypt the content that will be written to disc 25 using whatever encryption method host system 5 finds suitable. The associated security metadata (the security key(s)) required to decrypt the resulting encrypted content is shown as security data 200 in Figure 2. Using secure session key 60, security data 200 is encrypted in encryption block 205. Numerous encryption methods may be implemented in encryption block 205 – what is important is that, regardless of the encryption algorithm implied by security data 200, storage engine 20 must be able to decrypt the resulting encrypted security data 200 using secure session key 60 in decryption block 210. Using its own encryption method as denoted by storage engine security key 220, storage engine 20 then re-encrypts security data 200 in encryption block 230 and writes re-encrypted security data 200 to

disc 25. In the same fashion, storage engine 20 could also ~~re-encrypted~~ re-encrypt the associated encrypted content being received from host system 5. Note that the associated singly or doubly-encrypted content is not illustrated but would also be processed and written to disc 25 at this time as will be further explained herein.

Please replace the paragraph beginning on page 14, line 20 with the following replacement paragraph:

Those of ordinary skill in the art will appreciate that the encryption of security data 200 discussed with respect to Figure 2 is independent of the underlying content encryption schemes implemented ~~independently~~ in host system 5 and storage engine 20. In one embodiment of the invention, the content encryption (as opposed to the encryption of security metadata) provided by host system 5 may be the only encryption applied to the content to be secured. In other words, in such an embodiment, although storage engine 20 re-encrypts the encrypted security data 200 received from host system 5, it does not apply an analogous re-encryption to the associated encrypted content. Figure 3 illustrates the operation of a SECURE_WRITE_BLOCK command and a SECURE_READ_BLOCK command for a storage engine 20 that does not apply any further encryption to the encrypted content received from host system 5. To perform a write of a secure block of data to disc 25, host system 5 encrypts data 300 using security data 305 (because this encryption is applied to the content as opposed to encrypting the encryption key itself, this security metadata may also be denoted as a "content key"). Storage engine 20 writes the resulting encrypted content 310 to disc 25 using a conventional block write command process. The corresponding content key 305 is

encrypted/decrypted using secure session key 60 and then re-encrypted using a storage engine security key 220 according to a SECURE_WRITE_BLOCK command as described with respect to Figure 2. The resulting encrypted content key 305 is then written to disc 25. Similarly, in a read of encrypted content 300 from disc 25, storage engine 20 responds to a conventional block-level read command and provides encrypted content 310 to host system 5. Pursuant to a SECURE_READ_BLOCK command operation as described with respect to Figure 2, encrypted content key 307 is decrypted using storage engine key 220 and then encrypted with secure session key 60 within storage engine 20. After recovering content key 305 by decryption with secure session key 60, host system 5 may now decrypt encrypted content 310 to recover content 300.

Please replace the paragraph beginning on page 16, line 22 with the following replacement paragraph:

As discussed with respect to Figures 2 and 3, the present invention is independent of the type of encryption applied to the data content. A particular form of data encryption, however, will now be described with respect to Figures 4 and 5. Figure 4 illustrates steps in the execution of a SECURE_WRITE_BLOCK command in which host system 5 encrypts content 300 using an Advanced Encryption Standard (AES) algorithm. AES key generation module 400 generates the required AES key so that content 300 may be encrypted in AES encryption block 405. A secure session key 60 is generated by storage engine 20 as described with respect to Figures 1 and 2. Using secure session key 60, host system scrambles the AES key in AES key encryption block 410. Storage engine 20 receives the AES-encrypted content from host system 5 and

performs an additional encryption on this encrypted content using a Data Encryption Standard (DES) algorithm in encryption block 415. This encryption may occur using single, double, or triple DES encryption as described in U.S. Patent Application ~~Serial~~ No. 09/583,452, entitled "Method of Decrypting Data Stored on a Storage Device Using an Embedded Encryption/Decryption Means," filed May 30, 2000, the contents of which are hereby incorporated by reference. The corresponding DES key or keys are generated in DES key generation module 425. Using secure session key 60, data storage engine decrypts the AES key in AES key decryption block 420. The AES key is then DES-encrypted and the DES key(s) and the encrypted AES key are then stored in secure area 40 of disc 25. The associated doubly-encrypted data content is stored in file system area 85 of disc 25.

Please replace the paragraph beginning on page 16, line 22 with the following replacement paragraph:

In the embodiments described above, the storage engine 20 is relatively "dumb" in that it has no ownership of the file system used to with respect to the encrypted content on disc 25. For example, although storage engine 20 may have file-system-object-level access to security metadata in embodiments incorporating a security repository, storage engine 20 has no knowledge regarding the physical blocks a given file system object may occupy on disc 25. This knowledge is retained by host system 5, which must perform the translation of a file system object request into the corresponding block level request. In contrast to such an approach, U.S. Patent ~~Serial~~ Application No. 09/539,841, entitled "File System Management Embedded in a Storage Device," filed March 31, 2000, now

U.S. Patent No. 6,823,398, discloses a storage device that responds to file system object requests rather than block-level requests. Such a storage device controls where content is stored on the storage medium as well as the security data linkage. Similarly, a file system management system may be embedded within storage engine 20 such that a "hybrid file system management" results. In a hybrid file system embodiment of the present invention, storage engine 20 implements a file system for encrypted file system objects. At the same time, host system 5 maintains its own file system for non-secure file system objects. In hybrid file system embodiments, storage engine 20 will store file system objects and the associated security metadata in a manner of its own choosing. In this fashion, storage engine 20 may maintain tight control over its ability to link the security metadata with the encrypted file system objects. As discussed previously, there are a number or ways in which security metadata may be linked with encrypted files such as through streams. Regardless of how storage engine 20 implements this linkage in a hybrid file system embodiment, storage engine 20 should maintain a set of non-security-related metadata for each encrypted file system object such the file system object name, the file system object size, the physical blocks occupied by the file system object, attributes associated with the file system object such as having a read-only status, and dates and times for file system object creation and modification.